



JPW
AF

PATENT

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

Applicant: Finlay et al. Examiner: Anh Ly
Serial No. 09/628,599 Group Art Unit: 2172
Filed: July 28, 2000 Docket No. CA990018US1
Title: DIRECT CALL THREADED CODE

5

CERTIFICATE UNDER 37 CFR 1.8

I hereby certify that this correspondence and identified enclosures are being deposited with the United States Postal Service, first class mail, postage prepaid, under 37 C.F.R. § 1.8 on the date indicated below and is addressed to the Mail Stop: Appeal Brief-Patents, Commissioner for Patents, P. O. Box 1450, Alexandria, VA 22313-1450, on September 8, 2004.

10

Sandra Parker

APPEAL BRIEF

15 Mail Stop: Appeal Brief-Patents
Commissioner for Patents
P. O. Box 1450
Alexandria, VA 22313-1450

20 Sir:

This Brief is submitted pursuant to the Notice of Appeal filed on July 8, 2004 and is filed in triplicate, as required by 37 CFR Sec. 1.192.

25 The Commissioner is hereby authorized to charge payment of the \$330 Appeal Brief Fee due under 37 CFR Sec. 1.17(c) and any additional fees required for the above-identified application or credit any overpayment to Deposit Account No. 09-0460.

1. Real Party in Interest

30 The real party in interest in this appeal is International Business Machines Corporation of Armonk, New York and assignee.

2. Related Appeals and Interferences

There are no other appeals or interferences known to the appellant, the appellant's legal representative or the assignee which will directly affect or be directly affected by or have a bearing on the Board's decision in this appeal.

3. Status of Claims

Claims 1-22 remain pending in the application and are the subject of this appeal. Claims 1, 10 and 19 were previously amended on June 4, 2003 before an RCE was filed. A copy of all pending claims, including the claims on appeal, are set forth in an attached Appendix.

4. Status of Amendments

Response to Final Office Action containing a request for reconsideration submitted on June 2, 2004 has been entered, as indicated in the Advisory Action of July 15, 2004.

5. Summary of the Invention

As shown throughout the Specification, the present invention is directed to RDBMS with Interpreters, and specifically on pp. 3-4, it is shown that the invention relates to a system, program storage device and method for pre-processing an access plan generated for a query in a relational database management system to include a direct call mechanism replacing a lookup function of a run-time interpreter. The access plan includes a plurality of operation codes, each of the operation codes being associated with one or more executable functions for performing the query. Method includes the steps of:

(a) determining from the access plan an executable function associated with a first operation code; and

(b) augmenting said first operation code in the access plan with a pointer to said executable function to provide a direct call mechanism replacing a lookup function of a run-time interpreter.

The pre-pass mechanism of the present invention replaces the repeated looking up of the function to call to process the OPCODE and the function's operands, and any decisions that need to be made repeatedly (i.e. static decisions), during the interpreter phase of execution. The pre-pass mechanism comprises a pre-processing function which replaces or augments the OPCODE, and any static decisions, with a pointer to the function to call to perform the operation specified

by the OPCODE, or a pointer to an intermediate function with an auxiliary data structure, or a pointer to an auxiliary data structure, wherein the auxiliary data structure includes a pointer to the function to call to perform the operation specified by the OPCODE. Advantageously, the pointers are called without additional lookup. The intermediate function to call to perform the function specified by the OPCODE may include processing operations and static decision making.

Specifically, p. 2., li. 1-15 of Specification describe a creation of an access plan with op. codes, p. 2, li. 16-31 describe a prior art problem because an interpreter has to translate and run each op. code one at the time, and during that step interpretation it has to examine the op. code, look up an associated function which is called to process that op. code, make a decision in order to interpret and process the op. code and then run the op. code. On p.3, in the Summary section quoted above, it is shown that the present invention includes a pre-processing step which replaces each access plan op. code with a pointer to its function to provide a direct call, to avoid the lookup, wherein the pointer is named "a direct pointer". This is described in detail in pp. 7-9 of the Specification.

6. Issues

The issues on Appeal are as follows:

(A) Whether claims 1-22 are unpatentable under 35 U.S.C. Sec. 103(a) as being obvious over US Patent No. 6,438,536 issued to Edwards et al. (hereinafter Edwards) in view of US Patent No. 6,105,033 issued to Levine.

7. Grouping of Claims

The claims do not stand or fall together and arguments for patentability of each group of claims, as set forth, are set forth below.

Group I: claims 1-22, each of which stand or fall together.

8. Arguments

(A) Sec. 103(a) Rejection of Claims 1-22 (Group I)

All independent claims 1, 10 and 19 of the present invention are specifically directed to show an improvement of a standard database management system which includes implementation of a direct call mechanism replacing the lookup function of a run-time interpreter and a method for

pre-processing an already created access plan to provide a direct pointer and a direct call mechanism in such a system. They recite novel structure and thus distinguish over the cited prior art, under 35 U.S.C. 103(a). This is described in Figs. 1-4 and Specification on p. 2, li. 16-31; p. 3, li. 2-10, p. 6, li. 14-30, pages 7-9, and is used to provide faster access which is cost-effective.

5

Claim 1, for example, is directed to a method for pre-processing an access plan generated for a query in a relational database management system to include a direct call mechanism replacing a lookup function of a run-time interpreter, said access plan including a plurality of operation codes, each of said operation codes being associated with one or more executable functions for performing the query, said method comprising the steps of:

10

(a) determining from the access plan an executable function associated with a first operation code; and

(b) augmenting said first operation code in the access plan with a pointer to said executable function to provide a direct call mechanism replacing a lookup function of a run-time interpreter.

15

As can be seen, claimed method is for pre-processing of an existing access plan which has op. codes and is to be interpreted by a Software Interpreter (not a Compiler). Step (a) determines, for an op. code from the existing access plan, an associated executable function which would be interpreted if there is no present invention. Step (b) augments the op. code, inside the existing access plan, and disallows interpretation of the executable function by the Interpreter because it replaces the op. code with a pointer to the function, the function being located outside the access plan, which implements the operation indicated by the op. code, thus removing the interpretive step. This means that later on, when the access plan is being executed at run-time, instead of interpreting the op. code, the pointer is used to call the replacement code, thus a term "direct call" and "direct pointer", which is described in much more detail in the Specification.

20

25

30

Thus, the claims are directed to pre-processing, which happens prior to execution, which allows a run-time improvement of execution of an existing and previously optimized access plan whose steps have already been determined and are not changed by the present invention but are substituted with pointers (Spec. p. 7, li. 5-10) in order to increase the run-time speed. As can be seen on p. 7, li. 25, the pre-processing is performed by the Access Plan Manager prior to storing the improved access path plan into a memory cache and before the execution. As can be seen on

p.8, li. 14-16, this processed code section has pointers and is ready for execution, described on p. 8, li. 17-29.

i) Examiner has Misinterpreted the Limitations of the Claims and Prior Art

5

To establish prima facie obviousness of a claimed invention, all the claims limitations must be taught or suggested by the prior art, MPEP Sec. 2143.03, citing *In re Royka*, 180 USPQ 580 (CCPA 1974).

10 Applicant regrets that all his attorney's attempts to obtain an examiner's interview and explain misinterpretation of certain computer science terms in office actions were unnecessarily turned down, despite her numerous attempts to talk to the Examiner and his supervisors, thus prolonging prosecution and causing unnecessary costs. Each Response to Office Action clearly explained each misinterpretation but to no avail, because they were again repeated verbatim,
15 with all grammatical errors.

Moreover, the Advisory Action and Final Office Action are a verbatim repetition of the first two Office Actions, except for some new referenced lines. However, the same reference columns and lines were referred to with different argument, such as p. 9 reference to col. 6 and col. 8 lines.

20 Further, they also include all mistakes, which were pointed to in the Applicants' Responses, such as the Edwards' column number 45-67, which were not corrected. Moreover, they fail to follow the law, although repeatedly cited by the Applicant for Examiner's convenience.

Further, throughout all three Office Actions and Advisory Action the Examiner appears to argue
25 that because two references disclose the conventional terms well known and used in every write-up about database management systems that the references teach many elements in common and thus teach related art and can be combined to produce the claimed invention. Moreover, in order to satisfy rejection of just one limitation of a claim, the Examiner combines a few words from one column section with a few word from another column section with a few words from a third
30 column section with a few words from the forth column section. This gross misinterpretation of the law would deem every database invention obvious and unpatentable, because it is not the database term that is taught by the prior art but the action taken upon this item.

Applicant respectfully objects to the practice used to reject claims of the present invention because in each rejection in Office Actions a few lines of reference were cited and never a whole sentence or paragraph. Besides, the quoted language does not appear in the references and is taken verbatim from the claims of the present invention. Moreover, none of the references has even one whole element of the claims 1, 10 and 19 and their dependent claims. This was held in both Office Actions. Moreover, each reference misses many elements, as shown in both Office Actions, and is from a different field.

Applicant respectfully suggests that the term "pre-processing" of claims 1, 10 and 19 has been misunderstood by the Examiner in all three Office Actions and Advisory Action so that some cited prior art references which only teach query execution are used to reject claims of the present invention, which is from a different field. As shown in Specification and Figs. and all responses to Office Actions a pre-processing generates an optimized access plan. Query execution is performed afterwards.

Advisory Action further misquotes the claims when erroneously describing why the Office Actions do not give it patentable weight. Applicant respectfully rejects this explanation because, although the term "pre-processing" is placed in a preamble, it does not merely recite the purpose of a process nor the intended use of a structure because the body of the claim directly depends on the preamble for completeness because the claim 1 step (b) limitation of augmenting the first op. code has to happen during pre-processing stage, after creation of the access plan and before its execution. If the term is not used, the claim could be subject to misinterpretation, as was seen in all three Office Actions.

Further, the Advisory Action is mistaken regarding Applicant's use of terms "same executable function" and "direct pointer" because they are recited in rejected independent claims and Applicant's arguments should have not been arbitrarily ignored but addressed according to the law. Specifically, claim 1 limitation (a) explicitly defines "an executable function" and when limitation (b) refers to "said executable function" it is, according to the law, the same executable function, a limitation not shown in the prior art. Further, "a direct pointer" is recited in claim 1 (b), as "a pointer to provide a direct call mechanism" and is shown as a "direct pointer" throughout Specification and is not ambiguous at all. Last two paragraphs of Advisory Action

copy verbatim arguments from Office Actions with all grammatical errors pointed to by Applicant.

5 The section "Response to Arguments" of Final Office Action states that Examiner disagrees with Applicant and that Edwards "teaches executable function performing by lower component layers and during processing of generating code for a specific SQL query, the code generation component layer inserts calls to the difference performance enhancing subroutine in place of normally included calls to lower component layer. Subroutine pass pointers to the generation code for retrieval information to get results and a pointer to where the result should be returned
10 to the function" (see col. 7, li. 60-67, col. 8, li. 30-38 and abstract). Further, it states that "Edwards teaches pointers to functions where receive the returned result values of the calling or passing from the subroutines" (col. 8, li.28-48).

15 Applicant would like to clarify again some Computer Science terminology related to the present invention and cited references because there appears to be some confusion in Office Actions. A Compiler creates an executable file by compiling a whole program with all code lines at once; later, during execution time, the executable file code lines get executed. Present invention is directed to RDBMS with an interpreter and not a compiler. An Interpreter does not compile a whole program and does not create an executable file because it interprets and immediately
20 executes code lines one at the time. Further, there seems to be a confusion regarding "Function". A Function is a routine, as is a Subroutine and it is not data. They both receive input data and create output data (result), according to the inputted data. Therefore, a Function is code and not data. A Pointer can point to different structures; cited prior art references teach pointing to data (i.e., input or output parameters) which is not the same as pointing to a routine (Function) code
25 statement, as in the present invention.

Therefore, the Examiner's statements support Applicants' arguments from the Responses, further reiterated below, because they show that Edwards reference is applicable during execution and not during pre-processing, as is claimed in the present invention. Moreover, it inserts code to a
30 different subroutine and not to the same executable function routine, as is claimed in the present invention. Further, in Edwards, a pointer does not point to a routine (i.e., code as in Functions or Subroutines), as is claimed in the present invention, but to another type of pointer, a pointer to

output (i.e., data, such as result values). Therefore, Examiner's Response shows that the claims of the present invention are unobvious and are, thus, patentable.

Further misinterpretation of the fact and law is to use the Levine reference to render the present invention obvious because the Examiner's Response states that Levine teaches "The director component contains routines that generate calls for searching or looking up the information if the codes for those statements are found or for deleting the code segment" (col. 6, li. 12-26, col. 7, li. 1-30, col. 8, li. 20-62 and col. 14, li. 30-52. This section teaches away from the present invention because the claimed present invention replaces a lookup function of a run-time interpreter with a direct call so that looking up is avoided, which results in an improvement of speed.

Moreover, Col. 8, lines 15-19, 20-62, col. 6, li. 12-26, col. 7, li. 1-30 and col. 14, li. 30-52) of Levine are directed to the Cache Manager calls routines which provide a "code token" as an input which is "interpreted as a pointer" to a new, generated code segment or a data structure (referenced cols. 6-8 lines). Therefore it teaches away from the present invention because it does not replace the operation code with a pointer to the same executable function, as is claimed in the present invention. Moreover, it indirectly references not code but data locations (col. 9, li. 1-10) whereas the present invention uses a direct pointer which does not need interpretation. Further, Levine teaches Director calls to Cache Manager to search the cache for obsolete code and to call a new, Delete Obsolete Code function (col. 8, li. 15-63, col. 14, li. 30-53). Therefore, it does not replace the operation code with a pointer to the same executable function, as is claimed in the present invention. Thus, Levine teaches away from the present invention and it does not explicitly teach the claimed direct call mechanism replacing a lookup function of a run-time interpreter.

Therefore, Edwards and Levine references cannot be used to invalidate independent claims 1, 10 and 19, and their dependent claims because they fail to teach any and all the steps of these claims. Therefore, Examiner's Response shows that the claims of the present invention are unobvious and are, thus, patentable.

ii) Prima Facie Case of Obviousness by Prior Art Has Not Been Established and the Combination of Edwards and Levine Would Not Satisfy All the Limitations of the Claims

To establish prima facie obviousness of a claimed invention, all the claims limitations must be taught or suggested by the prior art, MPEP Sec. 2143.03, citing *In re Royka*, 180 USPQ 580 (CCPA 1974).

5 As stated in MPEP Sec. 706.02(j), 35 U.S.C. 103 authorizes a rejection where, to meet the claim, it is necessary to modify a single reference or to combine it with one or more other references. After indicating that the rejection is under 35 U.S.C. 103, the examiner should set forth in the Office action:

- 10 (A) the relevant teachings of the prior art relied upon, preferably with reference to the relevant column or page number(s) and line number(s) where appropriate,
(B) the difference or differences in the claim over the applied reference(s),
(C) the proposed modification of the applied reference(s) necessary to arrive at the claimed subject matter, and
(D) an explanation why one of ordinary skill in the art at the time the invention was made would have been motivated to make the proposed modification.

15 To establish a prima facie case of obviousness, three basic criteria must be met. First, there must be some suggestion or motivation, either in the references themselves or in the knowledge generally available to one of ordinary skill in the art, to modify the reference or to combine reference teachings. Second, there must be a reasonable expectation of success. Finally, the prior art reference (or references when combined) must teach or suggest all the claim
20 limitations. The teaching or suggestion to make the claimed combination and the reasonable expectation of success must both be found in the prior art and not based on applicant's disclosure. *In re Vaack*, 947 F.2d 488, 20 USPQ2d 1438 (Fed. Cir. 1991). See MPEP Sec. 2143 - 2143.03 for decisions pertinent to each of these criteria.

25 The initial burden is on the examiner to provide some suggestion of the desirability of doing what the inventor has done. "To support the conclusion that the claimed invention is directed to obvious subject matter, either the references must expressly or impliedly suggest the claimed invention or the examiner must present a convincing line of reasoning as to why the artisan would have found the claimed invention to have been obvious in light of the teachings of the references." *Ex parte Clapp*, 227 USPQ 972, 973 (Bd. Pat. App. & Inter. 1985). See MPEP Sec. 2144 - 2144.09 for examples of reasoning supporting obviousness rejections.

30 Office actions and Advisory Action are not following this law. Although Applicant argued in the Response to Office Action that the referenced prior art must be from the same field, since the references are obviously not solving the same problem, and that a combination or modification
35 must be shown in the prior art itself, each Examiner's Office Action failed to address these points and follow the law. As shown by the Applicant, the teachings of the referenced prior art are not relevant to the claimed invention, the proposed modifications of the applied reference(s) necessary to arrive at the claimed subject matter are not shown, and an explanation about why one of ordinary skill in the art at the time the invention was made would have been motivated to
40 make the proposed modification was never given.

Further, each cited reference is individually complete and they do not suggest a combination or modification and are impossible to combine. *Case Amgen, Inc. v. Chugai Pharmaceutical Co.*,

927 F.2d 1200, 18 USPQ2d 1016 Fed. Cir. 1991) is on point as is the case *In re Gordon*, 733 F.2d 900, 221 USPQ 1125 (Fed. Cir. 1984), which held that where there is no technological motivation for a modification or if a proposed modification of reference would destroy intent, purpose or function of the reference, the prima facie case of obviousness is not properly established. This law was not followed in the Office Actions either.

The Examiner has not established a prima facie case of obviousness because the three basic criteria stated above, which must be met, were not met because he did not point out: to any suggestion or motivation, either in the references themselves or in the knowledge generally available to one of ordinary skill in the art to modify the reference or to combine reference teachings, a reasonable expectation of success was not shown (and is impossible) and that the prior art reference(s), which must teach or suggest all the claim limitations, do so here, which they do not. Furthermore, the Examiner did not satisfy the initial burden to provide some suggestion in the references of the desirability of doing what the inventor has done, because to support the conclusion that the claimed invention is directed to obvious subject matter, either the references must expressly or impliedly suggest the claimed invention or the examiner must present a convincing line of reasoning as to why the artisan would have found the claimed invention to have been obvious in light of the teachings of the references.

Furthermore, Applicant repeatedly pointed out that the cited references are from nonanalogous art. MPEP Sec. 2141.01(a) on Analogous and Nonanalogous Art states:

TO RELY ON A REFERENCE UNDER 35 U.S.C. 103, IT MUST BE ANALOGOUS PRIOR ART

The examiner must determine what is "analogous prior art" for the purpose of analyzing the obviousness of the subject matter at issue. "In order to rely on a reference as a basis for rejection of an applicant's invention, the reference must either be in the field of applicant's endeavor or, if not, then be reasonably pertinent to the particular problem with which the inventor was concerned." *In re Oetiker*, 977 F.2d 1443, 1446, 24 USPQ2d 1443, 1445 (Fed. Cir. 1992). See also *In re Deminski*, 796 F.2d 436, 230 USPQ 313 (Fed. Cir. 1986); *In re Clay*, 966 F.2d 656, 659, 23 USPQ2d 1058, 1060-61 (Fed. Cir. 1992) ("A reference is reasonably pertinent if, even though it may be in a different field from that of the inventor's endeavor, it is one which, because of the matter with which it deals, logically would have commended itself to an inventor's attention in considering his problem."); and *Wang Laboratories Inc. v. Toshiba Corp.*, 993 F.2d 858, 26 USPQ2d 1767 (Fed. Cir. 1993).

PTO CLASSIFICATION IS SOME EVIDENCE OF ANALOGY, BUT SIMILARITIES AND DIFFERENCES IN STRUCTURE AND FUNCTION CARRY MORE WEIGHT

While Patent Office classification of references and the cross-references in the official search notes are some evidence of "nonanalogy" or "analogy" respectively, the court has found "the similarities and differences in structure and function of the inventions to carry far greater weight." *In re Ellis*, 476 F.2d 1370, 1372, 177 USPQ 526, 527 (CCPA 1973) *In re Clay*, 966 F.2d 656, 23 USPQ2d 1058 (Fed. Cir. 1992) (Claims were directed to a process for storing a refined liquid hydrocarbon product in a storage tank having a dead volume between the tank bottom and its outlet port wherein a gelled solution filled the tank's dead volume to prevent loss of stored product while preventing contamination. One of the references relied upon disclosed a process for reducing the permeability of natural

underground hydrocarbon bearing formations using a gel similar to that of applicant to improve oil production. The court disagreed with the PTO's argument that the reference and claimed inventions were part of the same endeavor, "maximizing withdrawal of petroleum stored in petroleum reserves," and found that the inventions involved different fields of endeavor since the reference taught the use of the gel in a different structure for a different purpose under different temperature and pressure conditions, and since the application related to storage of liquid hydrocarbons rather than extraction of crude petroleum. The court also found the reference was not reasonably pertinent to the problem with which the inventor was concerned because a person having ordinary skill in the art would not reasonably have expected to solve the problem of dead volume in tanks for refined petroleum by considering a reference dealing with plugging underground formation anomalies.).

The law cited above states that referenced art must either be in the field of applicant's endeavor or, if not, then be reasonably pertinent to the particular problem with which the inventor was concerned that logically would have commended itself to an inventor's attention in considering his problem. As is shown in MPEP citation above, *In re Clay* the court defined the endeavor very narrowly as just "a withdrawal of oil" which removes from the field of analogous art all other actions in petroleum industry. Further, it stated that the inventions involved different fields of endeavor since the reference taught the use of the gel in a different structure for a different purpose under different temperature and pressure conditions, and since the application related to storage of liquid hydrocarbons rather than extraction of crude petroleum, as shown in the reference. Thus, the court stated that actions of "storage" and "extraction" are different fields of endeavor and that purpose and conditions are also very important and it held that the reference was not reasonably pertinent to the problem with which the inventor was concerned.

The Office Actions do not cite any prior art which handles Direct Call Threaded Code. The present invention teaches pre-processing an access plan to provide a direct call mechanism replacing a lookup function of a run-time interpreter, as claimed in independent claims 1, 10 and 19 of the present invention. Edwards teaches dynamic code generation at query execution time. Levine teaches obsolete cache entries removal. Therefore, the references are from two different fields, none of which is related to a direct call mechanism replacing a lookup function of a run-time interpreter, as claimed in the present invention.

The methods taught by these two references are completely different of the present invention and are from different art fields. They cannot handle direct call mechanism to the same executable function but propose new code generation and use of cache delete function. Moreover, it is shown that they do not perform any elements of the independent claims 1, 10, and 19, and therefore their dependent claims. Further, they satisfy a different need from a different area and do not teach direct call mechanism. Therefore, these references cannot be used to invalidate

independent claims 1, 10, and 19 and their dependent claims. Because none of the referenced prior art teaches elements (a) to (b) of claims 1, 10 and 19, which are the main steps of the present invention, their combination is not a valid reason for rejection of these independent claims and claims dependent thereof. Therefore, each cited reference, by itself or in combination, cannot be used to invalidate claims 1, 10 and 19 because they fail to teach any and all the steps of these claims.

Claims 1-22 stand rejected under 35 U.S.C. 103(a) as being unpatentable over US Patent No. 6,438,536 issued to Edwards et al. (hereinafter Edwards) in view of US Patent No. 6,105,033 issued to Levine.

Sec. 103(a) Rejection of Claim 1 (Group I)

Examiner stated that, "with respect to claim 1, Edwards discloses determining from the access plan an executable function associated with a first operation code (see fig. 2B, the access plan is determined by the processing of optimizer component via execution functions associated with SQL statements as operation codes, which are generated by code generation component: col. 5, lines 32-64).

augmenting said first operation code in the access plan with a pointer to said executable function (the value of pointers in the operation code as a augment or parameter of a call function or subroutine, which is generated from SQL query code generation component or RAM CODEGEN executor: col. 4, lines 66-67 and col. 5, lines 1-10 and lines 40-56; also see col. 7, lines 60-67 and col. 8, lines 28-48, abstract)."

The Applicant respectfully objects to this misinterpretation because Edwards does not have the quoted language. It is the language taken verbatim from the claim 1 of the present invention, with Edwards reference column and line numbers added in brackets.

Examiner further stated that "Edwards discloses a query optimizer, Ram CODEGEN executor and code generation component for generating an access plan associated with the search queries entered by user and execution function such as call functions or call subroutines associated with SQL statements being as operation codes."

All independent claims 1, 10 and 19 of the present invention are specifically directed to show an improvement of a standard database management system which includes implementation of a direct call mechanism replacing the lookup function of a run-time interpreter and a method for pre-processing an already created access plan to provide a direct call mechanism in such a system. They recite novel structure and thus distinguish over the cited prior art, under 35 U.S.C. 103(a). This is described in Figs. 1-4 and Specification on p. 2, li. 16-31; p. 3, li. 2-10, p. 6, li. 14-30, pages 7-9, and is used to provide faster access which is cost-effective.

As discussed above, claimed method is for pre-processing of an existing access plan which has op. codes and is to be interpreted by a Software Interpreter (not a Compiler). Step (a) determines, for an op. code from the existing access plan, an associated executable function which should be interpreted if there is no present invention. Step (b) augments the op. code, inside the existing access plan, and disallows interpretation of the executable function by the Interpreter because it replaces the op. code with a direct pointer to the function, the function being located outside the access plan, which implements the operation indicated by the op. code, thus removing the interpretive step. This means that later on, when the access plan is being executed at run-time, instead of interpreting the op. code, the pointer is used to call the replacement code, thus a term "direct call", which is described in much more detail in the Specification.

Thus, the claims are directed to pre-processing, which happens prior to execution, which will allow a run-time improvement of execution of an existing and previously optimized access plan whose steps have already been determined and are not changed by the present invention but are substituted with pointers (Spec. p. 7, li. 5-10) in order to increase the run-time speed. As can be seen on p. 7, li. 25, the pre-processing is performed by the Access Plan Manager prior to storing the improved access path plan into a memory cache and before the execution. As can be seen on p.8, li. 14-16, this processed code section has pointers and is ready for execution, described on p. 8, li. 17-29.

It is true that Edwards in col. 4, li. 66-67, Fig. 2B, col. 5, li. 1-10, li 32-64, col. 7, li. 60-67 and col. 8, li. 28-48 and abstract mentions code generation component, CODEGEN executor, SQL query optimizer, pointers and call functions. These components are mentioned in most database applications. However, Edwards reference is from a different field and performs a different function. It is directed to a method used to dynamically generate different, performance

enhancing routines, which will, during query execution time, decide which execution routines to use, upon certain run-time conditions. This is shown in Abstract, col. 2, li. 50-61. According to col. 2, li. 62 to col. 3, li. 18, during code generation, the calls to these different, performance enhancing record file management (RFM) layer routines are inserted into the generated code, instead of normally included routine calls, in order to eliminate record management function layer. This is confirmed in col. 5, li. 1-11, col. 6, li. 40-52 and col. 7, li. 32-61. The term "during execution" is repeated numerous times, i.e., in col. 5, li. 8, col. 7, li. 32. Col. 7, li. 50-54 show that the enhances new routine decides to bypass the RFM layer and in col. 7, li. 58 the reference mentions that it is the "run-time decision". In col. 9, li. 1-3 it is shown that the reference reduces the number of calls to the RFM layer and I/O layer. In col. 10, li. 1-8 it confirms this and declares that the reference relates to systems that generate executable code at execution time. Further, pointer, as defined in col. 7, li. 60-67 and col. 8 li. 28-48 is not pointing to a function, as in the present invention, but to a part of a table row where data are stored.

Moreover, Edwards reference does not have any and all elements from the independent claims of the present invention. It does not include pointers to functions, does not use the normal executable function from an already generated access plan (step (a)), does not create a new access plan with pointers to the same executable function (step (b)) and is not directed to a pre-processing method replacing a lookup function of a run-time interpreter.

It is noted with appreciation that Examiner held that Edwards does not explicitly teach a direct call mechanism replacing a lookup function of a run-time interpreter. The Examiner held that Levine discloses code generation, from which operation codes are generated is having call functions routines, which are interpreted as a pointer to a generated code segment (col. 8, lines 20-62, col. 6, li. 12-26, col. 7, li. 1-30 and col. 14, li. 30-52).

Levine reference is from a different field of obsolete cache entries removal and has no elements of the claims of the present invention except code pointers. It is true that Levine reference uses pointers. A pointer is an often-used software tool which points to a code or data, depending on its function. However, in Levine reference a pointer points to the code to be deleted and not to be executed. Pointers in the present invention point to a function located outside the access plan which are to be executed. Col. 8, lines 15-19, 20-62, col. 6, li. 12-26, col. 7, li. 1-30 and col. 14, li. 30-52) of Levine are directed to the Cache Manager calls routines which provide a "code

token" as an input which is "interpreted as a pointer" to a new, generated code segment or a data structure (referenced cols. 6-8 lines) and does not replace the operation code with a pointer to the same executable function, as is claimed in the present invention. Moreover, it indirectly references data locations (col. 9, li. 1-10) whereas the present invention uses a direct pointer which does not need interpretation. Further, Levine teaches Director calls to Cache Manager to search the cache for obsolete code and to call a new, Delete Obsolete Code function (col. 8, li. 15-63, col. 14, li. 30-53) and does not replace the operation code with a pointer to the same executable function, as is claimed in the present invention. Thus, Levine teaches away from the present invention and it does not explicitly teach the claimed direct call mechanism replacing a lookup function of a run-time interpreter.

Further, the Examiner stated that it would have been obvious to a person of ordinary skill in the art at the time the invention was made to combine the teachings of Edwards with the teachings of Levine so as to obtain a call function for being interpreted in the execution time (Levine - col. 8, lines 15-19). This combination would have made the method for having code generation component to generate call function routines (Levine - Col. 8, lines 38-52) associated with execution functions and SQL statements (Levine - col. 3, lines 7-25) and during process of generating code for a specific SQL query, the code generation component produce calls to the difference performance enhancing into the generation code and this is easy to implement without having to make changes to the other components or layers of the relational database manager (Edwards - col. 45-67 and col. 3, lines 1-18).

The independent claims of present invention are not directed to "obtaining a call function for being interpreted in the execution time" but to use of pointers to avoid interpretation. They do not teach to "generate call function routines" because they already exist. The Levine reference, according to col. 8, lines 38-52, col. 3, li. 7-25 and throughout, teaches obsolete code deletion. Moreover, Edwards does not have col. 45-67 and col. 3, li. 1-18 explicitly teaches avoidance of the RFS layer, thus teaching away from the present invention. Moreover, these references are from completely different fields unrelated to a pre-processing method using pointers and the methods taught by these two references are among themselves from completely different art fields and cannot be combined. Moreover, it is shown that they do not perform any elements of the independent claims 1, 10, and 19 and therefore their dependent claims. Further, they satisfy a different need from a different area and do not teach pre-processing of existing access plan.

Therefore, these references cannot be used to invalidate independent claims 1, 10, and 19 and their dependent claims. Because none of the referenced prior art teaches elements (a) to (b) of claims 1, 10 and 19, which are the main steps of the present invention, their combination is not a valid reason for rejection of these independent claims and claims dependent thereof. Therefore, each cited reference, by itself or in combination, cannot be used to invalidate claims 1, 10 and 19 because they fail to teach any and all the steps of these claims.

Sec. 103(a) Rejection of Claim 2 (Group I)

Examiner stated that claim 2 stands rejected because Edwards discloses the remaining operation codes in the access plan (col. 5, lines 1-5 and lines 40-56; ; also see col. 2, lines 3-15).

The referenced language in Edwards is directed to prior art and query execution with enhanced functions, which is not claimed in claim 2 of the present invention, which mentions that steps from claim 1 are repeated for all steps in the access plan. Moreover, it is shown that this reference does not perform any elements of the independent claims 1, 10, and 19, and therefore their dependent claims. Further, it satisfies a different need from a different area and does not teach pre-processing of existing access plans to use pointers. Therefore, this reference cannot be used to invalidate the dependent claim 2.

Sec. 103(a) Rejection of Claims 3-4 (Group I)

With respect to claims 3-4, Examiner stated that Edwards discloses a method for pre-processing an access plan as discussed in claim 1 and that Edwards discloses a query optimizer, Ram CODEGEN executor and code generation component for generating an access plan associated with the search queries entered by user and execution function such as call functions or call subroutines associated with SQL statements being as operation codes.

However, as shown above, Edwards does not pre-process an existing access plan according to the independent claims of the present invention.

It is further held that Edwards does not explicitly teach a data structure for storing a pointer to said execution function and storing information associated with said executable function but that

Levine discloses a data structure that contains the pointers to the code segment (col. 8, lines 15-19, and lines 65-67 and col. 9, lines 1-12) and executable functions (col. 8, lines 40-52).

As shown above, Levine reference is from a different field of obsolete cache entries removal and has no elements of the claims of the present invention except pointers. It is true that Levine reference uses code pointers. However, col. 8, li. 15-19 of Levine is directed to the Cache Manager calls routines which provide a "code token" as an input which is "interpreted as a pointer" to a generated code segment or a data structure. The present invention uses a direct pointer which does not need interpretation. Thus, Levine teaches away from the present invention and it does not explicitly teach a direct call mechanism replacing a lookup function of a run-time interpreter. Col. 8, li. 65-67, col. 9, lines 1-12 and col. 8, lines 40-52 are directed to cache with indirect pointers and deletion of obsolete functions.

Further, the Examiner stated that it would have been obvious to a person of ordinary skill in the art at the time the invention was made to combine the teachings of Edwards with the teachings of Levine so as to obtain a data structure for pointers for allowing a process to indirectly reference locations within another data segment through pointers contained within the linkage segment (Levine - col. 9, lines 1-50, a call function for being interpreted in the execution time (Levine - col. 8, lines 15-19). This combination would have made the method for having code generation component to generate call function routines (Levine - col. 8, lines 38-52) associated with execution functions and SQL statements (Levine - col. 3, lines 7-25) and during process of generating code for a specific SQL query, the code generation component produce calls to the difference performance enhancing into the generation code and this is easy to implement without having to make changes to the other components or layers of the relational database manager (Edwards - col. 45-67 and col. 3, lines 1-18).

The independent claims of present invention are not directed to "obtaining a call function for being interpreted in the execution time" but to use of pointers to avoid interpretation. They do not teach to "generate call function routines" because they already exist. The Levine reference, according to col. 8, lines 38-52, col. 3, li. 7-25 and throughout, teaches obsolete code deletion. Moreover, Edwards does not have col. 45-67 and col. 3, li. 1-18 explicitly teaches avoidance of the RFS layer, thus teaching away from the present invention. Moreover, these references are from completely different fields unrelated to a pre-processing method using pointers and the

methods taught by these two references are among themselves from completely different art fields and cannot be combined. Moreover, it is shown that they do not perform any elements of the independent claims 1, 10, and 19 and therefore their dependent claims. Further, they satisfy a different need from a different area and do not teach pre-processing of existing access plan.

5 Therefore, these references cannot be used to invalidate independent claims 1, 10, and 19 and their dependent claims. Because none of the referenced prior art teaches elements (a) to (b) of claims 1, 10 and 19, which are the main steps of the present invention, their combination is not a valid reason for rejection of these independent claims and claims dependent thereof. Therefore, each cited reference, by itself or in combination, cannot be used to invalidate dependent claims 3
10 and 4 because they fail to teach any and all the steps of these claims.

Sec. 103(a) Rejection of Claim 5 (Group I)

With respect to claim 5, Examiner stated that Edwards discloses a method for pre-processing an
15 access plan as discussed in claim 1 and that Edwards discloses augmenting said first operation code in the access plan with a second pointer (the value of pointers in the operation code as a augment or parameter of a call function or subroutine: Col. 4, lines 66-67 and Col. 5, lines 1-10 and lines 40-56; also see Col. 7, lines 60-67 and Col. 8, lines 30-38).

20 Edwards does not have the quoted language mentioning second pointer. The referenced columns are described above and cannot be used to invalidate claim 5 of the present invention.

The Examiner further held that Edwards discloses a query optimizer, Ram CODEGEN executor and code generation component for generating an access plan associated with the search queries
25 entered by user and execution function such as call functions or call subroutines associated with SQL statements being as operation codes, and that Edwards does not explicitly teach a data structure for storing a pointer to said execution function and storing information associated with said executable function but that, however, Levine discloses a data structure that contains the pointers to the code segment (Col. 8, lines 15-19, and lines 65-67 and Col. 9, lines 1-12) and
30 executable functions (Col. 8, lines 40-52) and that, therefore, it would have been obvious to a person of ordinary skill in the art at the time the invention was made to combine the teachings of Edwards with the teachings of Levine so as to obtain a data structure for pointers for allowing a process to indirectly reference locations within another data segment through pointers contained

within the linkage segment (Levine - Col. 9, lines 1-50, a call function for being interpreted in the execution time (Levine - Col. 8, lines 15-19) and that this combination would have made the method for having code generation component to generate call function routines (Levine - Col. 8, lines 38-52) associated with execution functions and SQL statements (Levine - Col. 3, lines 7-25) and during process of generating code for a specific SQL query, the code generation component produce calls to the difference performance enhancing into the generation code and this is easy to implement without having to make changes to the other components or layers of the relational database manager (Edwards - col. 45-67 and col. 3, lines 1-18).

The referenced columns are described above and cannot be used to invalidate claim 5 of the present invention. Moreover, these references are from completely different fields unrelated to a pre-processing method using pointers and the methods taught by these two references are among themselves from completely different art fields and cannot be combined. Moreover, it is shown that they do not perform any elements of the independent claims 1, 10, and 19 and therefore their dependent claims. Further, they satisfy a different need from a different area and do not teach pre-processing of existing access plan. Therefore, these references cannot be used to invalidate independent claims 1, 10, and 19 and their dependent claims. Because none of the referenced prior art teaches elements (a) to (b) of claims 1, 10 and 19, which are the main steps of the present invention, their combination is not a valid reason for rejection of these independent claims and claims dependent thereof. Therefore, each cited reference, by itself or in combination, cannot be used to invalidate dependent claim 5 because they fail to teach any and all the steps of this claim.

Sec. 103(a) Rejection of Claim 6 (Group I)

Examiner stated that claim 6 stands rejected because Edwards discloses assessing the executable function associated with the first operation code and if applicable, replacing the call to the executable function with a call to a second executable function (col. 5, lines 6-10, col. 6, lines 40-45 and col. 10, lines 1-8).

The referenced language in Edwards is directed to prior art and query execution with enhanced functions which replace the RFM layer functions, which is not claimed in claim 6 of the present invention. Moreover, it is shown that this reference does not perform any elements of the

independent claims 1, 10, and 19, and therefore their dependent claims. Further, it satisfies a different need from a different area and does not teach pre-processing of existing access plans to use pointers. Therefore, this reference cannot be used to invalidate the dependent claim 6.

5 **Sec. 103(a) Rejection of Claim 7 (Group I)**

Examiner stated that claim 7 stands rejected because Edwards discloses intermediate function which includes processing operations for the first operation code or the executable function associated with the first operation code (col. 5, lines 5-32 and lines 40-55).

10

The referenced language in Edwards is directed to prior art and query execution with enhanced functions which replace the RFM layer functions, which is not claimed in claim 7 of the present invention. Moreover, it is shown that this reference does not perform any elements of the independent claims 1, 10, and 19, and therefore their dependent claims. Further, it satisfies a
15 different need from a different area and does not teach pre-processing of existing access plans to use pointers. Therefore, this reference cannot be used to invalidate the dependent claim 7.

Sec. 103(a) Rejection of Claims 8-9 (Group I)

20 Examiner stated that claims 8-9 stand rejected because Edwards discloses a method for preprocessing an access plan as discussed in claim 1 and that Edwards discloses a query optimizer, Ram CODEGEN executor and code generation component for generating an access plan associated with the search queries entered by user and execution function such as call functions or call subroutines associated with SQL statements being as operation codes but that
25 Edwards does not explicitly teach gathering statistics on the use of the executable function; and a pause for receiving user input before or after the call to the executable function and that, however, Levine discloses a analyzing each SQL statement entered by user for defining the access plan, which will produce the optimum performance for the execution of the statement (user enter queries into the database in order to obtain or extract requested data, and the query
30 optimizer analyzes how best to conduct the users' query of the database in terms of optimum speed in accessing the requested data: col. 5, lines 35-45 and lines col. 1, lines 25-30 and lines 38-45; also see col. 18, lines 28-32) and that, therefore, it would have been obvious to a person of ordinary skill in the art at the time the invention was made to combine the teachings of

Edwards with the teachings of Levine so as to obtain a way to analyze the SQL statement, which are entered by user (col. 5, lines 35-45 and col. 6, lines 15-25). This combination would have made the method for having routines for analyzing each query statement (Levine col. 6, lines 15-26), a code generation component to generate call function routines (Levine - col. 8, lines 38-52) associated with execution functions and SQL statements (Levine - col. 3, lines 7-25) and during process of generating code for a specific SQL query, the code generation component produce calls to the difference performance enhancing into the generation code and this is easy to implement without having to make changes to the other components or layers of the relational database manager (Edwards - col. 45-67 and col. 3, lines 1-18).

Claim 8 of the present invention is directed to gathering statistics on use of the executable function. Claim 9 of the present invention is directed to inserting a pause for receiving user's input.

It is noted with gratitude that the Examiner held that Edwards does not explicitly teach gathering statistics on the use of the executable function; and a pause for receiving user input before or after the call to the executable function. However, the fact analyzes each whole SQL statement does not mean that it teaches gathering statistics on the use of each one executable function. Moreover, the Office action is mute regarding inserting a deliberate pause for receiving user input before or after the call to the executable function but talks about inputting the whole SQL statement.

The referenced columns are described above and cannot be used to invalidate claims 8-9 of the present invention. Moreover, these references are from completely different fields unrelated to a pre-processing method using pointers and the methods taught by these two references are among themselves from completely different art fields and cannot be combined. Moreover, it is shown that they do not perform any elements of the independent claims 1, 10, and 19 and therefore their dependent claims. Further, they satisfy a different need from a different area and do not teach pre-processing of existing access plan. Therefore, these references cannot be used to invalidate independent claims 1, 10, and 19 and their dependent claims. Because none of the referenced prior art teaches elements (a) to (b) of claims 1, 10 and 19, which are the main steps of the present invention, their combination is not a valid reason for rejection of these independent claims and claims dependent thereof. Therefore, each cited reference, by itself or in combination,

cannot be used to invalidate dependent claims 8-9 because they fail to teach any and all the steps of these claims.

Sec. 103(a) Rejection of Claim 10 (Group I)

5

Examiner stated that Claim 10 is essentially the same as claim 1 except that it is directed to a computer program product rather than a method and is rejected for the same reason as applied to the claim 1 hereinabove.

10 Applicant respectfully points to his argument from above, regarding claim 1, 10 and 19, which shows that each cited reference, by itself or in combination, cannot be used to invalidate independent claims 1, 10 and 19 because they fail to teach any and all the steps of these claims.

Sec. 103(a) Rejection of Claim 11 (Group I)

15

Examiner stated that Claim 11 is essentially the same as claim 2 except that it is directed to a computer program product rather than a method and is rejected for the same reason as applied to the claim 2 hereinabove.

20 Applicant respectfully points to his argument from above, especially regarding claim 2 and claims 1, 10 and 19, which shows that each cited reference, by itself or in combination, cannot be used to invalidate independent claims 1, 10 and 19 and dependent claim 11 because they fail to teach any and all the steps of these claims.

Sec. 103(a) Rejection of Claim 12 (Group I)

25

Examiner stated that Claim 12 is essentially the same as claim 3 except that it is directed to a computer program product rather than a method and is rejected for the same reason as applied to the claim 3 hereinabove.

30 Applicant respectfully points to his argument from above, especially regarding claim 3 and claims 1, 10 and 19, which shows that each cited reference, by itself or in combination, cannot be

used to invalidate independent claims 1, 10 and 19 and dependent claim 12 because they fail to teach any and all the steps of these claims.

Sec. 103(a) Rejection of Claim 13 (Group I)

5 Examiner stated that Claim 13 is essentially the same as claim 4 except that it is directed to a computer program product rather than a method and is rejected for the same reason as applied to the claim 4 hereinabove.

10 Applicant respectfully points to his argument from above, especially regarding claim 4 and claims 1, 10 and 19, which shows that each cited reference, by itself or in combination, cannot be used to invalidate independent claims 1, 10 and 19 and dependent claim 13 because they fail to teach any and all the steps of these claims.

Sec. 103(a) Rejection of Claim 14 (Group I)

15 Examiner stated that Claim 14 is essentially the same as claim 5 except that it is directed to a computer program product rather than a method and is rejected for the same reason as applied to the claim 5 hereinabove.

20 Applicant respectfully points to his argument from above, especially regarding claim 5 and claims 1, 10 and 19, which shows that each cited reference, by itself or in combination, cannot be used to invalidate independent claims 1, 10 and 19 and dependent claim 14 because they fail to teach any and all the steps of these claims.

25 **Sec. 103(a) Rejection of Claim 15 (Group I)**

Examiner stated that Claim 15 is essentially the same as claim 6 except that it is directed to a computer program product rather than a method and is rejected for the same reason as applied to the claim 6 hereinabove.

30 Applicant respectfully points to his argument from above, especially regarding claim 6 and claims 1, 10 and 19, which shows that each cited reference, by itself or in combination, cannot be

used to invalidate independent claims 1, 10 and 19 and dependent claim 15 because they fail to teach any and all the steps of these claims.

Sec. 103(a) Rejection of Claim 16 (Group I)

5

Examiner stated that Claim 16 is essentially the same as claim 7 except that it is directed to a computer program product rather than a method and is rejected for the same reason as applied to the claim 6 hereinabove.

10 Applicant respectfully points to his argument from above, especially regarding claim 7 and claims 1, 10 and 19, which shows that each cited reference, by itself or in combination, cannot be used to invalidate independent claims 1, 10 and 19 and dependent claim 16 because they fail to teach any and all the steps of these claims.

15 **Sec. 103(a) Rejection of Claims 17-18 (Group I)**

Examiner stated that Claims 17-18 are essentially the same as claims 8-9 except that they are directed to a computer program product rather than a method and are rejected for the same reason as applied to the claims 8-9 hereinabove.

20

Applicant respectfully points to his argument from above, especially regarding claims 8-9 and claims 1, 10 and 19, which shows that each cited reference, by itself or in combination, cannot be used to invalidate independent claims 1, 10 and 19 and dependent claims 17-18 because they fail to teach any and all the steps of these claims.

25

Sec. 103(a) Rejection of Claim 19 (Group I)

Examiner stated that Claim 19 is essentially the same as claim 1 except that it is directed to a system rather than a method and is rejected for the same reason as applied to the claim 1
30 hereinabove.

Applicant respectfully points to his argument from above, regarding claims 1, 10 and 19, which shows that each cited reference, by itself or in combination, cannot be used to invalidate independent claims 1, 10 and 19 because they fail to teach any and all the steps of these claims.

5 **Sec. 103(a) Rejection of Claims 20-21 (Group I)**

Examiner stated that, with respect to claims 20-21, Edwards discloses a relational database system as discussed in claim 19, that Edwards discloses a query optimizer, Ram CODEGEN executor and code generation component for generating an access plan associated with the search
10 queries entered by user and execution function such as call functions or call subroutines associated with SQL statements being as operation codes. Edwards does not explicitly teach a data structure for storing a pointer to said execution function and storing information associated with said executable function, but, however, Levine discloses a data structure that contains the pointers to the code segment (col. 8, lines 15-19, and lines 65-67 and col. 9, lines 1-12) and
15 executable functions (col. 8, lines 40-52) and that, therefore, it would have been obvious to a person of ordinary skill in the art at the time the invention was made to combine the teachings of Edwards with the teachings of Levine so as to obtain a data structure for pointers for allowing a process to indirectly reference locations within another data segment through pointers contained within the linkage segment (Levine - col. 9, lines 1-50, a call function for being interpreted in the
20 execution time (Levine - col. 8, lines 15-19). This combination would have made the method for having code generation component to generate call function routines (Levine - col. 8, lines 38-52) associated with execution functions and SQL statements (Levine - col. 3, lines 7-25) and during process of generating code for a specific SQL query, the code generation component produce calls to the difference performance enhancing into the generation code and this is easy
25 to implement without having to make changes to the other components or layers of the relational database manager (Edwards - col. 45-67 and col. 3, lines 1-18).

Applicant respectfully points to his argument from above, especially regarding dependent claim 5 and claims 1, 10 and 19, which shows that each cited reference, by itself or in combination,
30 cannot be used to invalidate independent claims 1, 10 and 19 because they fail to teach any and all the steps of these claims and the dependent claims 20-21.

Sec. 103(a) Rejection of Claim 22 (Group I)

Examiner stated that, with respect to claim 22, Edwards discloses a method for pre-processing an access plan as discussed in claim 1, that Edwards discloses a query optimizer, Ram CODEGEN executor and code generation component for generating an access plan associated with the search queries entered by user and execution function such as call functions or call subroutines associated with SQL statements being as operation codes, that Edwards does not explicitly teach a data structure for storing a pointer to said execution function and storing information associated with said executable function but that, however, Levine discloses a data structure that contains the pointers to the code segment (col. 8, lines 15-19, and lines 65-67 and col. 9, lines 1-12) and executable functions (col. 8, lines 40-52) and that, therefore, it would have been obvious to a person of ordinary skill in the art at the time the invention was made to combine the teachings of Edwards with the teachings of Levine so as to obtain a data structure for pointers for allowing a process to indirectly reference locations within another data segment through pointers contained within the linkage segment (Levine - col. 9, lines 1-50, a call function for being interpreted in the execution time (Levine - col. 8, lines 15-19) and that this combination would have made the method for having code generation component to generate call function routines (Levine - col. 8, lines 38-52) associated with execution functions and SQL statements (Levine - col. 3, lines 7-25) and during process of generating code for a specific SQL query, the code generation component produce calls to the difference performance enhancing into the generation code and this is easy to implement without having to make changes to the other components or layers of the relational database manager (Edwards - col. 45-67 and col. 3, lines 1-18).

Applicant respectfully points to his argument from above, especially regarding dependent claim 5 and claims 1, 10 and 19, which shows that each cited reference, by itself or in combination, cannot be used to invalidate independent claims 1, 10 and 19 because they fail to teach any and all the steps of these claims and the dependent claim 22.

In view of the foregoing, it is submitted that the final rejections of claims 1-22 are improper and, accordingly, the Board is respectfully requested to reverse the final rejections and order that this application proceed immediately to issue.

iii) **The Examiner Uses Impermissible Hindsight to Modify the Teachings of Edwards Based on Levine in Order to Combine Them**

The mere fact that references can be combined or modified does not render the resulting combination obvious unless the prior art also suggests the desirability of the combination. See MPEP Sec. 2143.01, citing *In re Mills*, 916 F.2d 680, 16 USPQ2d 1430 (Fed. Cir. 1990). It is impermissible to use "hindsight reconstruction to pick and chose among isolated disclosures in the prior art to deprecate the claimed invention." *In re Fine*, 837 F.2d 1071, 5 USPQ2d 1596 (Fed. Cir. 1988).

Examiner relies on Edwards and Levine for disclosing all the limitations of the claims 1-22 but fails to point out how these quotations, even if existent in the cited prior art, can be combined or that the prior art shows desirability of the combination. The Examiner's position is that it would be obvious and he merely asserts it without any explanation.

As shown above, it is respectfully submitted that the prior art does not teach or even suggest modifying the teachings of Edwards and Levine to combine them. It is only the impermissible hindsight by the Examiner that the teachings can be combined and the Examiner has impermissibly used conventional terms for picking and choosing isolated disclosures in the prior art to assert that the claims are unpatentable. The cited prior art only discloses conventional database terms which would not lead a person of ordinary skill to use those unrelated terms and functions which do not relate to create the query pre-processing technique as claimed in the present invention. Moreover, it is impossible to combine the cited reference. Furthermore, most of these references teach against the present invention.

Since none of the references teach or suggest query pre-processing with a direct call mechanism, these cited references cannot be used to invalidate independent claims 1, 10 and 19 and their dependent claims because they fail to teach any and all the steps of these claims and they cannot be modified and combined. Thus, it is respectfully submitted that the cited prior art does not render claims 1-22 unpatentable.

9. Conclusion

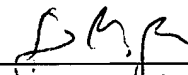
Regarding claims 1-22, none of the cited references teaches, shows, suggests or is even remotely related to pre-processing of existing access plans to augment op. code with pointers, as claimed by the present invention. Therefore, these reference cannot be used to invalidate independent claims 1, 10, and 19 and their dependent claims. Moreover, the Examiner combined references from different arts in order to reject claims 1-22, by quoting parts of sentences nonexistent in those references. However, even if these quotes are correct, the combination must be pointed to in the prior art itself and no such combination is pointed to in the cited references nor it could be since they are from different fields. Therefore, these references cannot be used to invalidate independent claims 1, 10 and 19 and their dependent claims because they fail to teach any and all the steps of these claims.

Improper combination of cited references is used in each claim rejection in the Office Action. None of the cited references suggests combination under In re Sernaker, 217 U.S.P.Q. 1, 6 (CAFC 1983), and one skilled in the art would have no reason to make a combination since they are from different fields, impossible to combine and individually complete. Moreover, none of the cited references discloses the subject matter and features of claims 1-22 of the present invention and even if they did show some individual features, they would not be able to meet the claims of the present invention which provide new and unexpected results over these references and are thus unobvious and patentable under Sec. 103(a).

In view of the foregoing, it is submitted that the final rejections of claims 1-22 are improper and, accordingly, the Board is respectfully requested to reverse the final rejections and order that this application proceed immediately to issue.

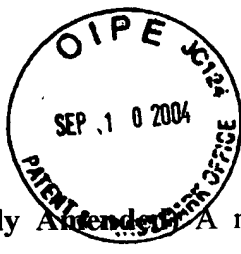
Respectfully submitted,

Date: September 8, 2004



Sandra M. Parker
Reg. No. 36,233

LAW OFFICE OF SANDRA M. PARKER
329 La Jolla Avenue
Long Beach, CA 90803
Phone/fax: (562) 597-7504



APPENDIX

1. (Previously Amended) A method for pre-processing an access plan generated for a query in a relational database management system to include a direct call mechanism replacing a lookup function of a run-time interpreter, said access plan including a plurality of operation codes, each of said operation codes being associated with one or more executable functions for performing the query, said method comprising the steps of:

(a) determining from the access plan an executable function associated with a first operation code; and

(b) augmenting said first operation code in the access plan with a pointer to said executable function to provide a direct call mechanism replacing a lookup function of a run-time interpreter.

2. The method as claimed in claim 1, further comprising repeating steps (a) and (b) for the remaining operation codes in the access plan.

3. The method as claimed in claim 1, wherein said step (b) comprises augmenting said first operation code in the access plan with a pointer to an intermediate function, said intermediate function including a data structure for storing a pointer to said executable function.

4. The method as claimed in claim 3, wherein said data structure includes means for storing information associated with said executable function or said first operation code.

5. The method as claimed in claim 1, wherein said step (b) comprises augmenting said first operation code in the access plan with a second pointer to a data structure, said data structure providing means for storing information associated with said first operation code or said executable function.

1 6. The method as claimed in claim 1, wherein said step (a) further includes assessing the
2 executable function associated with the first operation code and if applicable, replacing the call
3 to the executable function with a call to a second executable function.

1 7. The method as claimed in claim 3, wherein said intermediate function includes
2 processing operations for the first operation code or the executable function associated with the
3 first operation code.

1 8. The method as claimed in claim 7, wherein said processing operations in the intermediate
2 function include gathering statistics on the use of the executable function associated with the
3 operation code.

1 9. The method as claimed in claim 7, wherein said processing operations in the intermediate
2 function include a pause for receiving user input before or after the call to the executable
3 function.

1 10. **(Previously Amended)** A computer program product for use on a computer wherein
2 queries are entered by a user for retrieving data in a relational database management system
3 having a query optimizer for generating an access plan for executing the query, said query
4 optimizer including a direct call mechanism replacing the lookup function of a run-time
5 interpreter, said computer program product comprising:

6 a recording medium;

7 means recorded on said recording medium for instructing said computer to perform the
8 steps of:

9 (a) determining an executable function associated with a first operation code in the
10 access plan, the first operation code being one of a plurality of operation codes; and

11 (b) augmenting said first operation code in the access plan with a pointer to said
12 executable function to provide a direct call mechanism replacing a lookup function of a run-time
13 interpreter.

1 11. The computer program product as claimed in claim 10, the means for instructing said
2 computer further comprising repeating steps (a) and (b) for the remaining operation codes in the
3 access plan.

1 12. The computer program product as claimed in claim 10, wherein said step (b) comprises
2 augmenting said first operation code in the access plan with a pointer to an intermediate function,
3 said intermediate function including a data structure for storing a pointer to said executable
4 function.

1 13. The computer program product as claimed in claim 12, wherein said data structure
2 includes means for storing information associated with said executable function or said first
3 operation code.

1 14. The computer program product as claimed in claim 10, wherein said step (b) comprises
2 augmenting said first operation code in the access plan with another pointer to a data structure,
3 said data structure providing means for storing information associated with said first operation
4 code or said executable function.

1 15. The computer program product as claimed in claim 10, wherein said step (a) further

2 includes assessing the executable function associated with the first operation code and if
3 applicable, replacing a call to the executable function with a call to another executable function.

1 16. The computer program product as claimed in claim 12, wherein said intermediate
2 function includes processing operations for the first operation code or the executable function
3 associated with the first operation code.

1 17. The computer program product as claimed in claim 16, wherein said processing
2 operations in the intermediate function include gathering statistics on the use of the executable
3 function associated with the first operation code.

1 18. The computer program product as claimed in claim 12, wherein said processing
2 operations in the intermediate function include a pause for receiving user input before or after a
3 call to the executable function.

1 19. **(Previously Amended)** A relational database management system for use with a
2 computer system wherein queries are entered by a user for retrieving data from tables, the
3 relational database management system including a query optimizer for generating an access
4 plan associated with the queries entered by the user, said query optimizer including a direct call
5 mechanism replacing a lookup function of a run-time interpreter, said relational database
6 management system comprising:

7 (a) means for determining an executable function associated with each of a plurality
8 of operation codes in the access plan; and

9 (b) means for augmenting said operation codes in the access plan with a pointer to

10 said executable function associated with each operation code to provide a direct call mechanism
11 replacing a lookup function of a run-time interpreter.

1 20. The relational database management system as claimed in claim 19, wherein said means
2 for augmenting said operation codes includes means for replacing said operation codes in the
3 access plan with a pointer to an intermediate function, said intermediate function including a data
4 structure for storing a pointer to said executable function.

1 21. The relational database management system as claimed in claim 20, wherein said data
2 structure includes means for storing information associated with said executable function or said
3 operation codes.

1 22. The relational database management system as claimed in claim 19, wherein said means
2 for augmenting said operation codes includes means for adding another pointer to a data
3 structure, said data structure providing means for storing information associated with said
4 operation codes or said executable function.